Date=03/09/2020
Lecture By=Arkesh Jaiswal
Subject ⇒ Operating System-2

IN PREVIOUS LECTURE **(QUICK RECAP) Date-02/02/2020**	**In Today's Lecture (Overview)**
⇒ What is Process ⇒ Elements of Process ⇒ MCQs ⇒ Questions for self practice / Assignment for the Day	⇒ Threads - Linux Implementation ⇒ Fork and Exec ⇒ Context Switching ⇒ Copy on Write ⇒ The init process ⇒ MCQs ⇒ Questions for Self practice / CC for the day

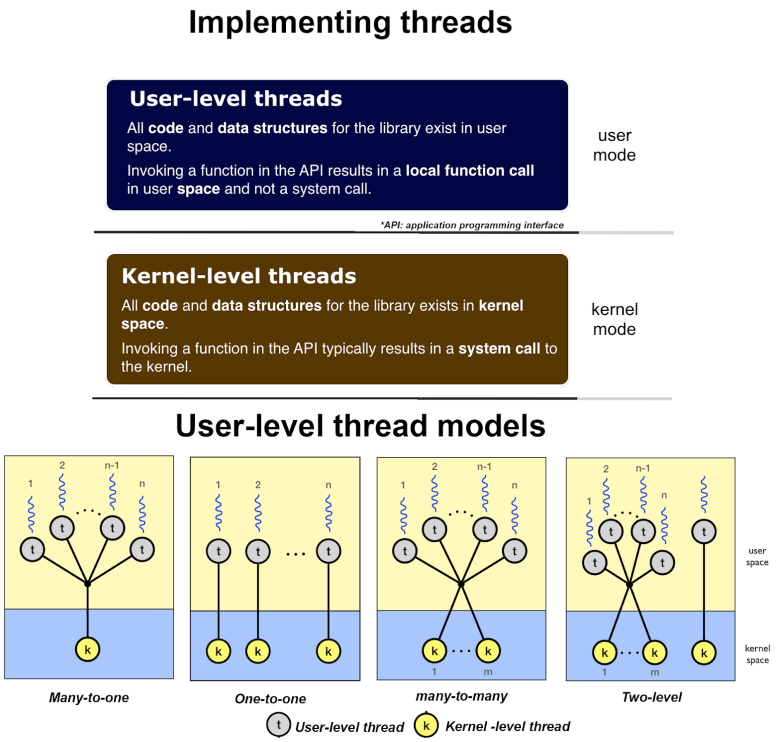
⇒ Threads - Linux Implementation

● The fork system calls copies everything including code and data

● This means the newly created process shares the same memory and code.

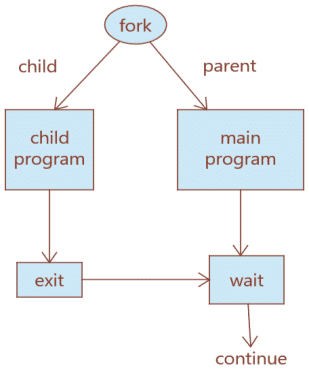
● This hybrid new process (child) is called a Thread.

● Threads have a number of advantages like
= Separate processes can not see each others memory
= Switching processes is quite expensive, and one of the major expenses is keeping track of what memory each process is using.



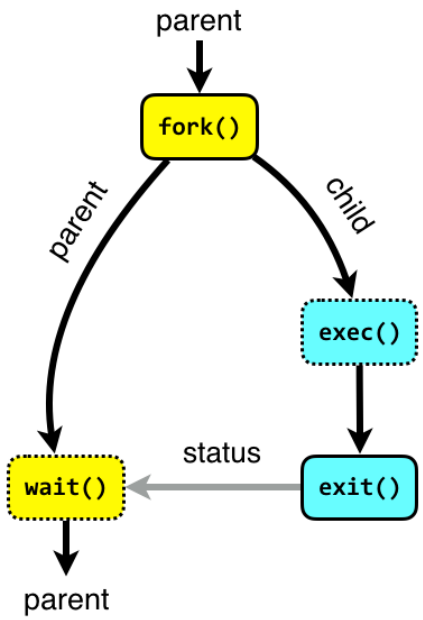
⇒ Fork and Exec

● New processes are created by the two related interfaces fork and exec.
● Fork - When OS sees the fork function call, create a new process that is exactly the same as the parent process.
● This means all the state like open files, register state and all memory allocations, which includes the program code etc are copied.



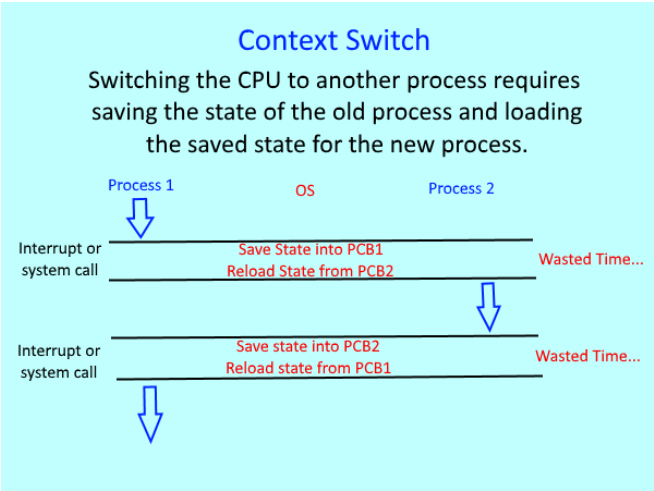
● Exec - Forking provides a way for an existing process to start a new one where as Exec provide a way to start a new process that is not part of the same program as

parent process.



⇒ Context Switching

● Context switching refers to the process the kernel undertakes to switch from one process to another.
● This allows multiple processes to share a single CPU, and is an essential feature of a multitasking operating system.
● A context is the contents of a CPU's registers and program counter at any point in time.



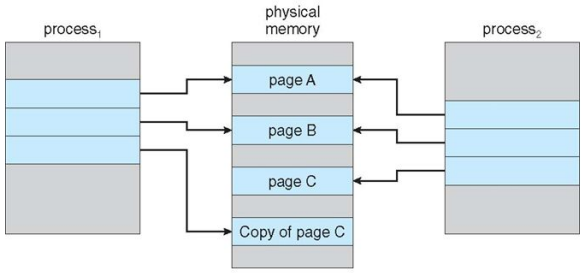
⇒ Copy on Write

● Copying the entire memory of one process to another when fork is called is an expensive operation.

● If the processes are only going to be reading the memory, then actually copying the data is unnecessary.

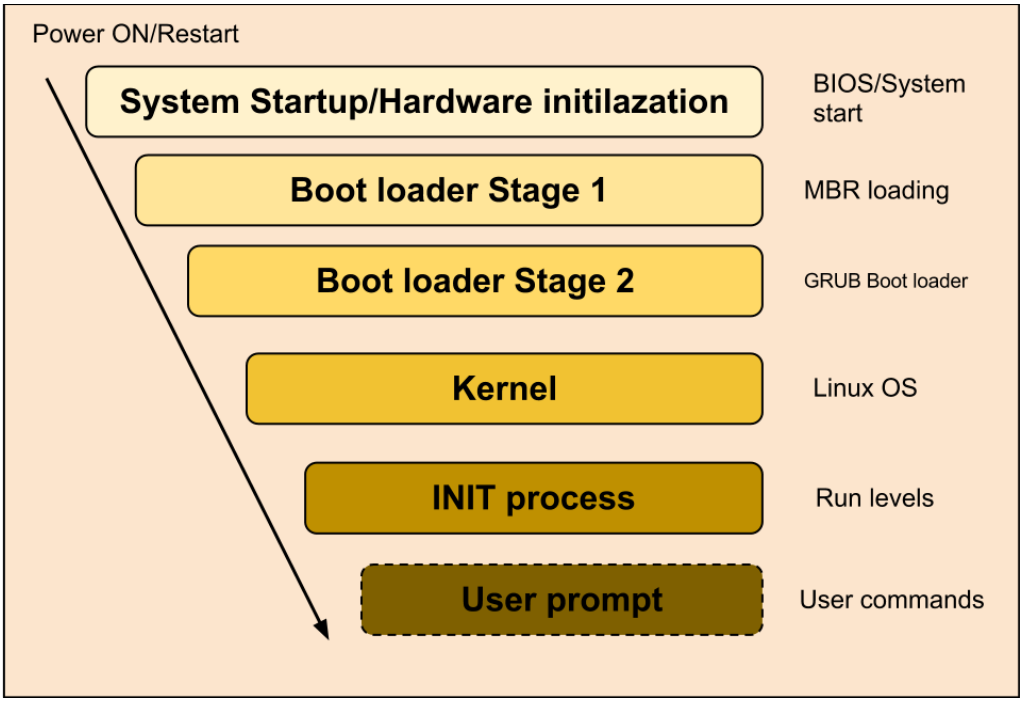
Copy on Write

- *Recall*: the `fork()` system call creates a child process that is a duplicate of its parent
- Since the child might not modify its parents pages, we can employ the copy-on-write technique:
  - The child initially shares all pages with the parent.
  - If either process modifies a page, then a copy of that page is created.



⇒ The init process

● On boot the kernel starts the init process, which then forks and execs the systems boot scripts.
● These fork and exec more programs, eventually ending up forking a login process.
● The process id for init is 0
● You can use pstree common on a Linux system to see the process hierarchy starting from init.



⇒ MCQs

1.**Which sys call creates a duplicate child process?**

A = fork()

B = exec()

C = open()

D = read()

2.**Which sys call stops the current process and starts a new one?**

A = fork()

B = exec()

C = open()

D = read()

3.Which process has an id 0?

A = init

B = boot

C = start

4.**What is the major hurdle for OS to optimize?**

A = multi threading

B = context switching

⇒ Questions for Self practice / CC for the day

⇒ **What are the components of a process?**